CS787: Advanced Algorithms **Homework 3**

Out: 3/1/22 **Due: 3/10/22**

1 Dynamic Updates in Flows

1. Suppose you have already computed the maximum flow in a network with m edges and integral capacities. Show how to update the maximum flow in $O(m)$ time after increasing the capacity of an edge by 1.
2. Suppose you have already computed the min-cost flow in a network with m edges and integral capacities and costs. Show how to update the min-cost flow in $O(mn)$ time after increasing the cost of an edge by 1.
   Hint: You may assume that the max-flow in any graph can be computed in $O(mn)$ time.
3. Let C be the maximum cost of any edge in a graph. Devise a cost scaling algorithm to compute the minimum cost flow from scratch, making $O(m \log C)$ calls to the algorithm in the previous part.

2 Minimum-Flow

The minimum flow problem is a close relative of the max flow problem with nonnegative lower bounds on edge flows (if an edge (i, j) has a lower bound $l_{ij} \geq 0$ and capacity u_{ij}, then the flow from i to j, f_{ij} must satisfy $l_{ij} \leq f_{ij} \leq u_{ij}$). In the minimum flow problem we wish to send a minimum amount of flow from the source s to the destination t while satisfying the given lower and upper bounds on edge flows f_{ij}.

1. Show how to solve the minimum flow problem by using two applications of any maximum flow algorithm that applies to problems with zero lower bounds on edge flows (e.g., the algorithms described in the lecture).
   Hint: first construct any feasible flow and then convert it into a minimum flow.
2. Prove the following min-flow-max-cut theorem. Let the lower bound on the cut capacity of an $s-t$ cut $(S, V \setminus S)$ be defined as $\sum_{(i,j)\in S\times(V\setminus S)} l_{ij} - \sum_{(i,j)\in(V\setminus S)\times S} u_{ij}$. Show that the minimum value of all feasible flows from node s to node t is equal to the maximum lower bound on cut capacity of all $s - t$ cuts.
3. A group of students wants to minimize their lecture attendance by sending only one of the group to each of n lectures. Lecture i begins at time a_i and ends at time b_i. It requires r_{ij} time to commute from lecture i to lecture j. Do not assume these times are integers. Develop a flow-based algorithm for identifying the minimum number of students needed to cover all the lectures.
   Hint: reduce to minimum flow.

3 Scheduling Meetings

It is visit day for the new graduate admits, and each wants a one-on-one visit with a certain set of faculty members. The day will be divided into time slots during which each student can meet at most one faculty member, and vice versa.

1. Suppose that the numbers of faculty and students are equal, each student wants to meet exactly d faculty, and each faculty member is on the request list of d students. Conclude that one can schedule a single slot in which every student is meeting someone.
   Hint: show the minimum cut of the desired-meetings graph is large.
2. Conclude that it is possible to schedule all the meetings to take place in d time slots.
3. Consider an arbitrary set of desired meetings. Obviously one needs at least as many slots as there are faculty to meet a given student, and students to meet a given faculty. Prove that one can arrange all meetings with no more slots than this number s.

4 Exploring Madison

You bought a new car and you want to drive around in the streets of Madison. It is in bucket list to visit every single street at least once. The map is given as a directed graph $G = (V, E)$ where every edge e has a positive length l_e. Your goal is to find the shortest route that starts and ends at the same node and visits every edge at least once. You are allowed to visit every edge or every node as many times as you like but want to minimize the total length of your route.

You know that a route exists that visits all edges exactly once if and only if the out-degree of each node is equal to its in-degree (Eulerian graph). This is true by Euler's theorem and there exists an efficient algorithm to find it. However, this is not the case for your graph, and you need to visit some edges more than once. Devise an algorithm to compute the shortest route by reducing the problem to min-cost matching and the problem of finding an Euler tour in a balanced degree graph.

Hint: what edges must be added to make the graph Eulerian?